VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Systém pro správu projektů

Project Management System

2019 Dušan Kučeřík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Dušan Kučeřík**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Systém pro správu projektů
Project Management System

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je vytvořit webovou aplikaci pro správu projektů postavenou na frameworku Angular. Součástí práce bude přehled existujících řešení pro online správu projektů. Aplikace bude umožňovat správu klientů a k nim přidružených projektů. U každého projektu bude možné definovat úkoly, chyby a další náležitosti spojené s řízením projektů. Aplikace bude také umožňovat komunikaci s klienty v podobě issue tracking systému.

Hlavní body zadání:
1. Přehled a srovnání existujících řešení v této oblasti.
2. Přehled použitých knihoven a nástrojů (představení jazyka TypeScript a Angularu).
3. Návrh a implementace aplikace pro správu projektů na platformě .NET (na straně serveru) a Angular frameworku (na straně klienta).
4. Shrnutí dosažených výsledků.

Seznam doporučené odborné literatury:

[1] STOREKHEYROLLAHI, Tugberk Ugurlu; Alexander Zeitler; Ali. Pro ASP.NET Web API: HTTP Web Services in ASP.NET. New York, NY: Apress, 2013. ISBN 978-143-0247-258.
[2] FREEMAN, ADAM Pro Angular. APRESS L.P., 2017. ISBN 1484223063.
[3] FENTON, Steve. TypeScript For JavaScript Programmers. lulu.com, 2013. ISBN 1291107371.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Janoušek**

Datum zadání: 01.09.2018
Datum odevzdání: 30.04.2019

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 26. dubna 2019

.....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 26. dubna 2019

.....

Děkuji svému vedoucímu práce Ing. Janu Janouškovi za odborné a metodické vedení, ochotu a pomoc v průběhu vypracovávání práce.

Abstrakt

V této bakalářské práci se budu věnovat problematice "Systému pro správu projektů". V první části popíšu jaká existují řešení v této problematice a pár vybraných ukážu a porovnám s ostatními. V další části budou představeny jazyk TypeScript a framework Angular a následně popsány knihovny, které jsou v bakalářské práci použity. Jako další bude navržena a naimplementována webová aplikace pro správu projektů na platformě .NET na straně serveru a Angular frameworku na straně klienta. Ke konci budou shrnuty dosažené výsledky.

Klíčová slova: správa projektů, Angular, TypeScript, systém, webová aplikace

Abstract

In this bachelor thesis I will analyze the issue of "Project Management System". The first part contains information on solving problems in this issue and comparison between selected five of existing systems. In the next part, there will be introduced programming language TypeScript and Angular framework, which are used in the bachelor thesis. In the next part, a web application for managing projects on the platform will be designed and implemented. At the end, the results will be summarized.

Key Words: project management, web application, system, Angular, TypeScript

Obsah

Seznam použitých zkratek a symbolů

TS	–	Programovací jazyk TypeScript
JS	–	Programovací jazyk JavaScript
URL	–	Uniform Resource Locator - adresa zdroje
API	–	Application Programming Interface - rozhraní pro programování aplikací
HTTP	–	Hypertext Transfer Protocol - internetový protokol určený pro komunikaci s WWW servery
HTML	–	Hypertext Markup Language - značkovací jazyk používaný pro tvorbu webových stránek
CSS	–	Cascading style sheets - kaskádové styly, popis způsobu zobrazení HTML elementů
DOM	–	Document Object Model - objektově orientovaná reprezentace HTML nebo XML dokumentu
MVC	–	Architektura Model-view-controller
SQL	–	Strukturovaný jazyk pro práci s daty v databázích
MSSQL	–	Microsoft SQL server
JSON	–	JavaScript Object Notation - zápis dat, určený pro přenos dat
JWT	–	JSON Web Token
SHAxxx	–	Rodina hashovacích algoritmů
CRUD	–	Čtyři základní operace nad databází (Create, Read, Update, Delete)
ORM	–	Objektově relační zobrazení
SMTP	–	Internetový protokol určený pro přenos emailů
mac OS	–	Operační systém Applu pro počítače
iOS	–	Operační systém Applu pro mobilní zařízení

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

V posledních letech stoupá požadavků zvýšení kvality práce za co nejmenší dosažitelnou dobu. V dnešní hektické době lze tedy ve velkých týmech zapomenout na část práce nebo jí špatně rozvrhnout, což zapříčiní časové zpoždění nebo třeba nestihnutí dokončení práce do uzávěrky. Přesně pro takové týmy a firmy vzniká mnoho systému pro správu projektů, které mají tento problém efektivně pomoci řešit či úplně vyřešit. Nejen že dobře navržený systém dokáže intuitivně a efektivně zobrazovat informace o daných projektech, ale také například usnadňuje komunikaci v rámci týmu a se zákazníky. Je tedy jasné, že přínos těchto systémů pro správů projektů týmům, při řešení problémů, je daleko větší, než kdyby takový problém řešil tým samostatně bez využití systému.

Dnes je takových systémů na trhu kvantum, tedy každý tým nebo firma si dokáže přijít na své, jelikož každý systém nabízí v podstatě ty samé možnosti. Nicméně téměř každý systém nabízí vymoženosti se zaměřením ve specifických oborech práce, ať už se jedná o informační technologie nebo třeba ekonomicky zaměřený obor.

Důvodem pro výběr práce je rozšíření obzorů v problematice systémů pro správu projektů, osvojení si nástrojů a technologií použitých při implementaci systému a prohloubení znalostí.

Cílem práce je tedy zaprvé přehled a srovnání již existujících systémů pro správů projektů, které lze využít ať už pro samostatné využití nebo jako využití pro celé týmy a malé firmy. Druhým cílem práce bude takový systém navrhnout a implementovat do funkční podoby. Práce bude tedy obsahovat základní popsání a představení nástrojů a programovacích jazyků použitých při implementaci systému. Navržený systém bude probrán jak po programátorské části, tak po části uživatelské a jejího grafického zpracování.

Jelikož je v práci mnoho popsaných metod a nástrojů z informačních technologií a technologií pro tvorbu webu a webových aplikací, tak je pro správné pochopení doporučeno, aby čtenář měl alespoň minimální znalosti v těchto technických odvětvích.

2 Přehled a srovnání existujících řešení v této oblasti

Systému pro správu projektů, které mohou použít jednotlivci, firmy či korporáty je v dnešní době nespočet. Počet těchto systémů den ode dne roste a zvyšuje se konkurence pro již existující řešení. Většina těchto systémů sdílí kvantum společných funkcí, vychytávek a možností, které vycházejí už s definice samotného systému pro správu projektů - mezi tyto patří například plánování projektu a k němu přidružených úkolů, plánování pracovního plánu na delší časové období či správu uživatelů v podobě zaměstnanců nebo klientů, kteří mají možnost vzdáleně kontrolovat prováděnou práci na svých zakázkách.

Systémy, které přicházejí na trh až nyní se snaží nabídnout něco víc oproti svým předchůdcům a proto zde vidíme stálé zlepšení v nabízení těchto služeb. Dnes už se tyto systémy předhánějí jen v tom, který nabídne zákazníkovi vyšší rychlost, dostupnost, jednoduchost ve smyslu práce se systémem a nabídne větší možnosti uchovávání informací v podobě, jakou zákazník potřebuje.

Většina těchto systému nabízí jak webové rozhraní pro přístup do systému přes webový prohlížeč, tak programy či aplikace, které lze spustit na mnoha zařízeních. Těmito zařízeními myslíme například počítače s operačním systémem Windows nebo Mac OS. Podpora pro Linux je ještě stále velice malá a systémů, které podporující právě Linux či jeho další distribuce (Ubuntu, Mint, Fedora atp.) je stále omezený oproti svým konkurentům od Microsoftu a Applu. Dalšími zařízeními, kterými lze v dnešní době již dost často přistupovat do takového systému jsou nepochybně chytré telefony. Jelikož chytré telefony jsou, řekněme na vzestupu, tak systémů, které podporují chytré telefony je taktéž pouze omezené množství. Avšak počet systémů s podporou pro chytré telefony nezabrzditelně roste enormní rychlostí při pokusu konkurovat ostatním systémům. Zde se můžeme setkat s podporou pro chytré telefony se systémy iOS, Android a Windows Phone.

Níže bude představeno 5 vybraných systému pro správu projektů. U každého budou uvedeny jeho plusy a mínusy, výhody a nevýhody oproti ostatním a jednoduše popíšu orientaci v nich samotných, co obsahují a jaké nabízejí možnosti ve smyslu přístupu k nim, podpoře a podobně.

Všechny níže představené systémy nabízejí zaučení ke správnému používání systému pomocí webinářů, online chatu přímo ze systému nebo dokumentace, kde nabízejí velké množství návodů a frekventovaných otázek.

2.1 Wrike.com

Wrike je velmi sofistikovaný nástroj pro správu projektů online, který nabízí aplikace pro Mac OS, Windows, iOS i Android. Je ideální pro středně velké a podnikové společnosti, které přijímají práci od mnoha zákazníků, mají mnoho projektů a klientů. Wrike umožňuje zákazníkovi si jej přizpůsobit podle jeho potřeb pomocí vlastních pracovních postupů, polí a zpráv. Hlavní mise tohoto systému je, aby udělal týmy mnohokrát produktivnější a dohlížel, aby vše bylo splněno v předem stanoveném časovém rozmezí. Další poslání Wrike je usnadnit, zefektivnit a udržet transparentnost každodenní práce pro jeho zákazníky a zaměstnance.

Mezi hlavní výhody tohoto systému patří:

1. Již při registraci se systém uživatele táže v jakém oboru se budou jeho projekty pohybovat. Je tomu tak, neboť systém zákazníkovi při prvním přihlášení navrhne pár šablon z daného oboru, které by mohly uživateli pomoci při tvorbě projektů či úkolů. Uživatel si v tomto systému může udělat vlastní šablony projektů či úkolů, které se často opakují, aby nemusel pokaždé vytvářet téměř ten samý úkol či projekt.

2. Pracovní prostředí nabízí položku "Dashboards", kterou můžeme chápat jako nějakou domácí obrazovku. Narozdíl od jiných systémů, kde je tato obrazovka často předem definovaná, si ji může uživatel systému Wrike plně přizpůsobit svým potřebám. K tomu slouží "Widgety". Systém nabízí kolem 15 předem definovaných šablon widgetů (například "Úkoly mi přiřazené", "Mé úkoly po splatnosti", "Dnešní úkoly", "Přehled aktivity u projektu"atp.), ke kterým si uživatel může přiřadit projekt/úkol dle jeho uvážení nebo si dokonce widget sám vytvořit pomocí nabízených filtrů.

3. Navigační panel na levé straně obsahuje stromovou strukturu složek a projektů v nich. Je to tedy velmi přehledný systém, kde složku můžeme chápat jako jeden celek více projektů a projekty jako celek úkolů. Jednotlivé složky/projekty/úkoly se dají oštítkovat pro ještě lepší organizaci. Jako štítky se můžou používat i samotné projekty nebo úkoly pomocí jednoduchého přetažení.

4. V nastavení si každý uživatel může vybrat barevné téma celého systému z již předem definovaných barevných kombinací.

5. Wrike dále nabízí různé rozšíření pro firmami používané služby jako Outlook/Gmail, Google Chrome, G Suite nebo třeba Github.

2.2 monday.com

monday.com je určen pro jednotlivce nebo týmy všech velikostí, pomáhá jak malým týmům ve větší produktivitě, tak velkým podnikům pracovat jako tým a lépe uspořádat práci. Obdobně jako u předešlého systému Wrike, se i monday.com při registraci bude zajímat v jakém oboru systém bude pracovat a jaké části daného oboru bude uživatel spravovat s daným systémem (předdefinované jsou například tickety, požadavky, problémy, sprinty.

Mezi hlavní výhody tohoto systému patří:

1. Svým uživatelům nabízí personalizovanou URL adresu, pro rychlý přístup k danému systému. Výhodou je snadné sdílení adresy. Tato adresa je pro uživatele dostupná v podobě "https://nazev.monday.com/", kde v URL místo položky "nazev"přidáme předem definovaný název uživatelského systému při registraci.

2. Podobně jako u předešlého systému Wrike.com, i zde je možnost si domácí obrazovku uživatele všemožně personalizovat. Personalizace je dosaženo taktéž pomocí tzv. "widgetů", tudíž uživatel ihned po přihlášení zde vidí přehled na daný den nebo například minulý týden. Vše závisí na uživateli samém jak si tuto stránku personalizuje.

3. Na plánování projektů má trochu jiný pohled než ostatní systémy a většina systému je tvořena jistými deskami, které můžeme chápat jako tabulku podobnou tabulce, kterou známe z tabulkových procesorů různých Office balíčků. Desky obsahují přímo úkoly (v systému nazvané jako jisté "Pulses") nebo skupiny. Do skupin lze opět řadit úkoly pro lepší přehlednost.

4. Úkoly v podobě tabulky mohou mít uživatelem definované sloupce, které si uživatel může plně personalizovat. Od barev buněk po popis celých sloupců přímo stavěných pro jeho potřeby.

5. Systém obsahuje tři typy desek.

   (a) Hlavní typ je takový, ke kterému mají přístup všichni uživatele přiřazení do týmu celého systému.

   (b) Sdílený typ je typ desky určený pro klienty. Do vstupu stačí zadat emailové adresy klienta/klientů, kteří mají tento přístup dostat. Klient se v tuto chvíli stává jakýmsi uživatelem pouze pro čtení, tudíž má možnost kontrolovat průběh práce na jeho zakázce.

   (c) Posledním typem desky je typ soukromý. Taková deska může být určena pouze pro jejího majitele nebo pro uživatele, které její majitel pozve ke spolupráci.

6. Další výhodou monday.com je možnost synchronizovat uživatelův kalendář například s Google Calendar.

7. Jelikož je systém a jeho projekty/úkoly chápán podobně jako tabulkový procesor, tak nabízí uložení celé desky (projektu) do tabulky například pro MS Office Excel. Jednotlivé desky lze uložit jako šablonu a při dalším vytváření podobného projektu tuto desku použít pro rychlejší a jednodušší integraci projektu do systému.

2.3 ActiveCollab

ActiveCollab je pokročilý systém pro správů projektů, který je vysoce konfigurovatelný, bohatý v plánování a nabídce organizačních nástrojů pro urychlení procesu vykonávání práce nebo automatizace plateb pro projekty. Podobně jako předchozí systémy nabízí již při registraci šablony podle vybraného oboru (například "Návrh webových stránek", "Logo design"atp.). Obdobně jako u předešlých systému se navigační menu nachází na levé straně obrazovky, odkud můžeme přistupovat k různým systémovým položkám jako jsou stránky s přehledem projektů, mé práce (zde

se nacházejí úkoly od daného uživatele, čas strávený v úkolech strávený prací a přehled záznamů každé aktivity v systému uživatele). Další položkou je velmi přehledný kalendář s časovou osou, kde lze jednoduchým přetahováním myši přemisťovat úkoly z jednoho časového okna na jiné. Kalendář se mimo jiné podobně jako u monday.com dá synchronizovat s osobním kalendářem uživatele, například Google Calendar.

Mezi hlavní výhody tohoto systému patří:

1. Nabízí možnost nechat systém generovat faktury, kde lze změnit jazyk a měnu podle země, ze které zákazník pochází a ihned ji zákazníkovi zaslat z prostředí systému.

2. Jako již dříve zmíněné systémy, i tento nabízí aplikace pro operační systémy Mac OS a Windows nebo operační systémy chytrých telefonů iOS a Android. Jelikož systém dokáže generovat faktury a odhady, je zde i podpora PayPal, Apple Pay či kreditních karet.

3. Narozdíl od monday.com, kde zákazník může přistoupit k projektu pomocí URL za účelem sledování práce na jeho zakázkách, ActiveCollab nabízí zaregistrovat klienta, který má dokonce možnost pod svým účtem vytvářet úkoly ve svých zakázkách, které jsou poté zpracovány zaměstnancem týmu.

4. Pokud má uživatel nějaké problémy nebo nejasnosti, tak je zde nabízena live podpora, kde lze jednoduše komunikovat se zaměstnancem ActiveCollab prostřednictvím real time chatu.

5. U každého projektu nebo úkolu je nabízena real time diskuze (oproti výše uvedeným systémům, kde byla možnost diskuze, ovšem ne real time diskuze). Ke každému projektu/úkolu lze přidávat poznámky, výdaje a zaregistrovat množství času strávené na daném projektu.

6.

2.4 Zoho Projects

Zoho Projects je cloudově-založený systém pro správu projektů navržen pro malé a středně velké společnosti nebo týmy. Pomáhá docílit výsledků v předem stanoveném časovém rozmezí a zjednodušuje práci jednotlivým zaměstnancům. Řešení tohoto softwaru je postaveno tak, aby zlepšilo proces nasazení projektu od počáteční fáze až po dokončení s použitím předem stanovených a automatizovaných funkcí. Stejně jako systémy Wrike a monday.com nabízí tento systém možnost personalizovat si domovskou obrazovku pomocí předem nadefinovaných karet, ovšem na rozdíl od svých konkurentů nelze vytvářet své vlastní karty s personalizovaným obsahem. Na počáteční šabloně se nachází mnoho grafů, na kterých lze vidět produktivitu v grafické podobě a například lze vidět práci ve formě úkolů. V horní liště se nachází projekty a položka pro přidání obsahu k projektu bez nutnosti projekt navštívit. Lze přidávat soubory, úkoly, statusy či třeba začít konverzaci. Po rozkliknutí projektů a výběru jednoho z nich se do horní lišty přidají právě rozkliknuté projekty pro rychlý přístup k nim a na levé straně je k dispozici navigační menu pro

daný projekt. Navigační menu na levé straně lze plně personalizovat a například projektů vůbec nemusíme přiřazovat možnost přidávat úkoly, zpřístupnit chat nebo zakázat kalendář. Každý projekt nabízí svou vlastní domovskou stránku, kde lze najít všechny přehledy. U každého projektu je také k dispozici real time chat, přehled ve formě kalendáře s časovou osou, soubory k němu přiřazené či třeba předem definované milníky.

Mezi hlavní výhody tohoto systému patří:

1. Velmi zajímavou funkcí, kterou Zoho Projects nabízí se nazývá Gamescope. Jedná se o funkci, která zlepší produktivitu zaměstnanců v týmu tím, že zaměstnanci za každý posunutý milník, splnění úkolu nebo třeba pokroku získávají body. V každé hře se dá nastavit časové rozmezí a také vybrat zaměstnance, se kterými chcete hrát.

2. V časové ose lze nastavit závislosti mezi jednotlivými úkoly abychom definovali pořadí práce. To znamená, že například druhý úkol potřebuje první úkol k tomu, aby se na něm začalo pracovat a pokud tento první úkol není hotový, tak na druhém nelze začít.

3. Další výhodou je možnost přidávat soubory k projektům ve stromové struktuře. Oproti ostatním výše uvedeným systémům se tímto práce se soubory stává o mnoho přehlednější. Systém nabízí nahrát soubor, který lze potom jednoduše přiřadit do více projektů najednou.

4. Nabízí real time chat, který lze využívat napříč celé webové aplikace nebo desktopovému klientu. Nicméně je zde i možnost chatu v daném projektu či přímo u úkolu.

5. Systém podporuje aplikace pro Android a iOS, ovšem oproti ostatním konkurentům stále nenabízí aplikace pro Windows nebo Mac OS.

2.5 Asana

Asana je webová a mobilní aplikace navržena tak, aby pomohla týmům a společnostem organizovat, sledovat a spravovat jejich práci. Pomáhá spravovat projekty a úkoly v jednotném prostředí pomocí webové aplikace, mobilních aplikací pro chytré telefony s operačními systémy iOS a Android nebo pomocí klientů v podobě aplikací na operačních systémech Mac OS a Windows. Asana nabízí plnou synchronizaci s většinou aplikací od Googlu, podporu pro Slack, Microsoft Office 365 a pro ukládání a sdílení souborů u projektům lze využít například Dropbox. Na levé straně se nachází navigační panel, na kterém se nachází mé úkoly, schránka s notifikacemi a všechny projekty, ke kterým jsem přiřazen.

Mezi hlavní výhody tohoto systému patří:

1. Asana umožňuje uživateli vytvářet šablony, které lze později použít pro podobné či totožné projekty. K nim umožňuje přidávat vlastní štítky, aby byly relevantní k projektu. Pokud uživatel nemá vlastní šablony pro projekt/úkol, tak je zde možnost využít šablon od Asany podle oboru jaký uživatel zvolil při registraci.

2. Celý systém je navržen mnohem intuitivněji a jednodušeji, aby uživatel nemusel tápat co a jak kde udělat. Z tohoto minimalistického pohledu vychází i domovská obrazovka, kde nemáme žádné karty s přehledy, nicméně vidíme zde jen úkoly uživatele, které musí splnit do uzávěrky. Pokud by uživatel potřeboval podobné přehledy jako u předchozích systému, lze toho docílit v takzvaných portfoliích, kde lze vidět pokrok všech projektů, jejich datum dokončení, kdo je k nim přiřazen a například jejich status.

3. Každý projekt nabízí záložku "Pokrok", kde jednotliví uživatelé přiřazeni k projektu mohou informovat své vedoucí o tom, jak jsou na tom s úkoly nebo pokud mají například problém s řešením dané věci.

4. Ostatně stejně jako výše zmíněné systémy, i tento nabízí chat u každého projektu, nicméně nelze komunikovat v rámci úkolů.

2.6 Srovnání vybraných systémů pomocí tabulky

Níže bude uvedena tabulka pro lepší přehled nabízených funkcí jednotlivých výše popsaných systémů. Funkcí a možností, které tyto systémy obsahují jsou kvanta a většinu těchto funkcí a možností již téměř všechny systémy podporují. Proto v Tabulce 1 bude uvedeno pouze několik málo možností, kde jsou tyto systémy rozdílné.

Tabulka 1: Srovnání základních nabízených možností vybraných systémů pro správu projektů [10, 11]

Možnost	Wrike	monday.com	ActiveCollab	Zoho Projects	Asana
Ideální počet uživatelů	10+	2+	1+	1+	2+
Personalizované šablony	✗	✓	✓	✓	✓
Podpora Mac OS a Windows	✓	✓	✓	✗	✓
Podpora iOS a Android	✓	✓	✓	✓	✓
Podpora 24/7	✓	✓	✗	✓	✗
Osobní zaučení systému	✗	✗	✗	✓	✓

3 Představení jazyka TypeScript

Typescript je open-source programovací jazyk vydáván pod licencí Apache 2.0, který vznikl pod záštitou firmy Microsoft v roce 2012, přesněji 1. října. Poslední verze k 1. dubnu je verze 2.3.3, který vyšla 22. května 2017. TypeScript je nadmnožinou JavaScriptu, který zahrnuje celý jazyk JavaScript a k tomu přináší soubor dalších užitečných funkcí. Mezi hlavní užitečné funkce patří podobně objektově-orientovaný syntax jazyka jako .NET, který se následně kompiluje zpět do JavaScriptu, který dokážou zobrazit dnes již všechny podporované webové prohlížeče.[2] To znamená, že každý JavaScriptový kód je sám o sobě také platným TypeScript kódem. To dělá programy napsané v TypeScriptu přenosné téměř na všech počítačích – například ve webových přohlížečích, na webu a nativních aplikacích v operačních systémech. TypeScript dále rozšiřuje JavaScript o statické typování a atributy, které známe z objektově orientovaného programování jako jsou třídy, dědičnost, moduly a další. Další výhodou skutečnosti, že TypeScript vychází z JavaScriptu je, že programátor může využívat již existující knihovny napsané v JavaScriptu jakými jsou například jQuery. Jelikož TypeScript vyvíjí sám Microsoft, tak lze očekávat, že zde existuje vynikající podpora pro jazyk TypeScript uvnitř vývojového prostředí Visual Studio a Visual Studio Code, které také pocházejí od Microsoftu. Nicméně podporu pro vývoj tohoto jazyka přidaly i ostatní společnosti do svých vývojových prostředí/textových editorů jako například Atom, Sublime text nebo Vim.

3.1 Základ TypeScriptu

Jak je uvedeno výše, TypeScript je nadmnožinou JavaScriptu, který zahrnuje celý programovací jazyk JavaScript a přidává další užitečné funkce, oproti tomu zde existují i podmnožiny JavaScriptu, které mají za cíl osekat JavaScriptu dostupné funkce a tím vytvořit menší jazyk pro daný úkol.[1]
Jelikož je TypeScript nadmnožinou JavaScriptu, tak je potřeba u něho myslet na to, že všechny standardní struktury obsaženy v JavaScriptu jsou okamžitě k dispozici i v programu TypeScriptu. Jedná se například o:

1. Typy dat
2. Operátory
3. Řízení toku a podmíněné příkazy (if, loops..)
4. Aritmetické, logické a další funkce

3.2 Proměnné

Proměnné v jazyce TypeScript musí splňovat a dodržovat pravidla pojmenování, které jsou použity u JavaScriptu. Identifikátor pro pojmenování proměnné musí splňovat následující podmínky:
První znak může obsahovat

1. Velké a malé písmeno
2. Podtržítko na začátku - používá se pro jednoduché značení privátních proměnných
3. Znak měny $ - používá s pro indikaci že je proměnná Observable (vysvětleno v kapitole o Angularu)
4. Unicode znaky

Proměnné jako takové mají funkční rozsah. To znamená, že pokud jsou proměnné deklarovány na nejvyšší úrovni daného programu, tak jsou k dispozici globálně. Takových proměnných by nemělo být mnoho z důvodu rizika kolizí při pojmenování dalších proměnných. Naopak proměnné deklarované uvnitř funkcí jsou takzvané funkční proměnné, ke kterým lze přistupovat pouze v rozsahu funkce. Stejně jako funkční proměnné lze deklarovat proměnné v dalších kontextech, tím mám na mysli například ve třídách – ty se poté nazývají jako třídní proměnné. [1, 2]

3.3 Datové typy

TypeScript nabízí statické typování. To znamená že typová kontrola probíhá automaticky aby bylo zabráněno neplatnému přiřazení hodnot. Pokud si ovšem programátor nepřeje použít statické typy, lze využít primitivní datové typy – rovnou přidělit proměnné datový typ, který nám zajistí aby nebyla přiřazena nesprávná hodnota tomuto datovému typu. Mezi primitivní datové typy se v TypeScriptu řadí tyto tři:

1. string – obsahuje sekvenci znaků
2. boolean – může mu být přiřazena pouze hodnota true nebo false
3. number – obsahuje číselnou hodnotu

TypeScript nenabízí žádný datový typ pro reprezentaci integeru nebo jiných specifických variant čísel z toho důvodu, že by nebylo praktické staticky kontrolovat jestli jsou všechny možné typy přiřazeny správně.[2] TypeScript dále nabízí datový typ „any“. Tento datový typ se používá pokaždé, když TypeScript nedokáže nebo mu nelze odvodit daný datový typ nebo pokud chce programátor explicitně tento dynamický datový typ použít. Tomuto datovému typu se poté dá přiřadit jakýkoliv z primitivních typů, rozhraní nebo tříd. Typový systém obsahuje ještě další 3 typy, které se ovšem nepoužívají na anotaci typu ale spíše odkazují na absenci hodnot. Mezi tyto 3 typy patří typ „undefined“, který je hodnotou proměnné, kterému ještě nebyla přiřazena hodnota. Dalším typem je typ „null“, který reprezentuje umyslnou absenci hodnoty objektu. A třetím a posledním typem je typ „void“, který se používá pouze pro návratový typ funkcí, které nevrací žádnou hodnotu nebo typ argumentu pro generickou třídu nebo funkci.[1]

3.4 Pole

Pole v jazyce TypeScript lze deklarovat u každého datového typu pouhým přidáním hranaté závorky za název typu. Při přidání položky do pole dochází ke kontrole datového typu, zda-li souhlasí s předem deklarovaným datovým typem daného pole. Pokud ovšem programátor deklaruje pole s datovým typem any, vypne tím kontrolu datových typů jak bylo zmíněno výše a tudíž takové pole může obsahovat prvky různých typů, například čísla, objekty, funkce, další pole nebo primitivní datové typy. Pole takového typu ovšem nese různé nevýhody - například nemožnost seřadit pole pomocí nativní funkce. Pro přístup k požadovanému prvku v poli lze přistoupit pomocí hranatých závorek s číslem uvnitř, které udává daný index pole. Pole se obdobně jako v dalších programovacích jazycích indexují od 0, nikoliv od 1. Tudíž k prvnímu prvku v poli lze přistoupit pomocí čísla 0 v hranatých závorkách. Narozdíl od například C#, pole v TypeScriptu nejsou omezeny délkou neboť mají proměnnou délku. Počet položek v poli se dá získat pomocí vlastnosti „length“, přidání a odebrání položek probíhá pomocí metod push() pro přidání a pop() pro odebrání prvku z pole.

```
// Typy deklarace polí

var emptyArray: any[];
var emptyArray: any[] = [];
var emptyArray: any[] = new Array();

//Do pole lze ihned inicializovat data, jednotlivé data oddělujeme čárkou

var notEmptyArray: string[] = ["Josef", "David", "Lukáš"];
```

Výpis 1: Typy deklarace polí v jazyce TypeScript [1, 2]

3.5 Rozhraní

Rozhraní neboli anglicky a v jazyce samotném *Interface*, lze chápat jako popis proměnné, která bude obsahovat data nebo metody v požadovaném tvaru. Jinými slovy lze rozhraní nazvat jako stavební jednotky. Rozhraní se dá použít jako datový typ, který má uvnitř další vlastnosti, které pokaždé při použití daného rozhraní musí být použity. Respektive žádnou z deklarovaných vlastností daného rozhraní není možné vynechat. Zároveň nám rozhraní deklaruje, jaké datové typy tyto vlastnosti mají. V rozhraních lze deklarovat metody, nicméně jejich definice probíhá až ve třídě, která dané rozhraní implementuje.[2] Taková definice metody rozhraní je povinná a ve třídě, která jej implementuje, dochází ke kontrole zda-li daná třída obsahuje všechny metody deklarované v rozhraní. Rozhraní deklarujeme pomocí klíčového slova interface. Rozhraní mohou

obsahovat vlastní konstruktor, který se deklaruje pomocí klíčového slova new() a také může obsahovat metody.

3.6 Třídy

Obdobně jako například v .NET i zde v TypeScriptu můžeme seskupit vlastnosti, metody a události do tříd reprezentující objekty. Je to jeden z hlavních aspektů pro plně fungující koncept objektově-orientovaného programování. Každá třída lze vytvořit, vytvořit instance třídy, volat metody třídy a manipulovat s událostmi dané třídy.
JavaScript je již delší dobu schopný objektově-orientovaný jazyk.Nicméně způsob jakým JavaScript podporuje objektově-orientované programování je velmi divný, kdežto TypeScript přichází se známou podobou jakou známe například z C#.

```
class SimpleClass {
   //metody, proměnné
}
```

Výpis 2: Vytvoření třídy v TypeScriptu

TypeScript kód je tedy mnohokrát jednodušší na pochopení, vypadá jednodušeji a čistěji než zkompilovaný kód do JavaScriptu, který lze vidět níže.

```
var SimpleClass = (function () {
   function SimpleClass() { }
      return SimpleClass;
})();
```

Výpis 3: Zkompilovaný kód TypeScriptu do JavaScriptu - Vytvoření třídy[1]

1. **Konstruktor** – Základní stavební jednotkou třídy v TypeScriptu je bezpochyby konstruktor. Každá třída obsahuje implicitní konstruktor, který programátor může přepsat explicitním konstruktorem. Je to metoda, která se volá při vytvoření instance dané třídy. Při volání této metody dochází k případné inicializaci všech stavů objektu.

2. **Modifikátory přístupu** – Všechny datové typy, vlastnosti a metody obsažené v třídě vždy obsahují modifikátor přístupu, ať už se jedná o implicitní nebo explicitní. Pokud programátor nedefinuje jinak, tak ve výchozím nastavení jsou vlastnosti a metody veřejné – není tedy potřeba přidávat klíčové slovo public. K takovým vlastnostem a metodám lze přistupovat zvenčí po vytvoření instance dané třídy. Nicméně pokud programátor bude chtít vlastnost či metodu skrýt, nebo omezit pouze pro třídní použití, lze takto učinit přidáním klíčového slova private. TypeScript nabízí ještě jeden modifikátor přístupu –

jedná se o modifikátor protected. Takový modifikátor přístupu lze použít v případě, kdy programátor bude chtít zamezit přístupu z venku, jelikož se chová stejně jako modifikátor přístupu private s jednou výjímkou tříd, které dědí z dané třídy.

3. **Metody a vlastnosti tříd** – Vlastnosti tříd se typicky deklarují ještě před konstruktorem. Jakmile je program kompilován, tak se inicializace těchto vlastností přesouvá do konstruktoru, tudíž při každém vytvoření nové instance dané třídy jsou vždy k dispozici každé této instanci. Ke třídním vlastnostem, proměnným a metodám lze přistupovat pomocí klíčového slova this. Lze taktéž deklarovat statické proměny, ke kterým se přistupuje pomocí názvu třídy a nikoliv pomocí klíčového slova this. Je tomu tak, neboť statická proměnná není definovaná pro každou instanci třídy.

4. **Dědičnost tříd** – Další výhodou objektově-orientovaného programování je zajisté možnost dědit vlastnosti a metody z jiných tříd nebo rozhraní. V TypeScriptu máme tedy dvě možnosti dědičnosti.
   1. Třída může implementovat rozhraní pomocí klíčového slova implements. Třída může implementovat více než jedno rozhraní, pouze při implementaci stačí mezi rozhraní přidat čárku. Například: *class MyClass implements Interface1, Interface2* [2]
   2. Třída může dědit z další třídy pomocí klíčového slova extends. Potomek získá všechny vlastnosti a metody od rodiče. Dané metody jdou dále v potomkovi přepisovat, pokud je zde potřeba rozdílné definice.[2]

3.7 Generika

Generika v TypeScriptu podobně jako u ostatních programovacích jazyků, které generika podporují, nabízí možnost psát algoritmus tak, aby datový typ prováděných operací mohl být přiřazen až později když to bude potřeba. Výhoda generik je v tom, že algoritmus stačí napsat jednou s tím, že datový typ se přiřadí později a programátor nemusí psát stejný algoritmus vícekrát pro větší počet datových typů. TypeScript podporuje tvorbu generických funkcí včetně generických metod, rozhraní a tříd.
Generika vytvoříme tak, že ihned za jméno funkce přidáme parametr do ostrých závorek (<>). S tímto typem pracujeme v těle funkce. Níže je uveden příklad pro lepší pochopení tvorby takového generika.

```
//Generická třída
class MainId<T> {
   constructor(public id: T) {
   }

   //Generická metoda
```

```
    get value(): T {
        return this.id;
    }
}
```

Výpis 4: Příklad vytvoření generické třídy a metody[1]

3.8 TypeScript operátory

Operátory definují funkce, které budou provedeny na datech. Data na kterých operátory používáme se poté nazývají operandy. Operátory jako takové dále dělíme na aritmetické, logické, přiřazovací, relační, ternární neboli podmiňovací a bitové.

Tabulka 2: Přehled operátorů nabízených jazykem TypeScript [8]

Operátor	Význam a popis
++, –	Přírůstek a snížení
+, -, *, /, %	Přičítání, odečítání, násobené, dělení a zbytek po dělení
<, <=, >, >=	Méně než, méně než nebo rovno, více než, více než nebo rovno
==, !=	Rovnost nebo nerovnost
===, !==	Test zda-li se jedná o ten samý typ
&&, ||	Zároveň a nebo
=	Přiřazení hodnoty k proměnné
+ (řetězec)	Spojování řetězců

3.9 Shrnutí kapitoly

V této podkapitole bude shrnuto ve zkratce vše, co bylo napsáno v celé kapitole Základy TypeScriptu.
- Celý TypeScript je technicky vzato JavaScript
- Při pojmenovávání proměnných se používá _ pro privátní proměnnou, $ pro observable
- Mezi primitivní typy patří integer, boolean, string a pro dynamické typy se uvádí datový typ any
- Pole v TypeScriptu nejsou omezeny délkou neboť mají délku proměnnou.
- V rozhraní lze deklarovat metody, ovšem nelze je definovat. Definici těla funkce má na starosti až třída, která takové rozhraní implementuje.
- Konstruktor je metoda, která se zavolá při každém vytvoření instance dané třídy.
- TypeScript nabízí 3 modifikátory přístupu ve třídách, jedná se o: public, private, protected
- Třídy mohou implementovat rozhraní nebo dědit z jiných tříd jejich vlastnosti a metody.

- Pomocí generik můžeme v TypeScriptu napsat například metodu pro více datových typů pouze jednou s tím, že přiřazení typu se provede až později.

4 Představení frameworku Angular

Angular je open-source framework založen na programovacím jazyku TypeScript vyvíjen a sponzorován firmou Google. Jelikož je open-source, tak je vyvíjen i komunitou a dalšími velkými korporacemi. Framework je softwarová struktura, která pomáhá při programování tím, že je předem naprogramovaná a obsahuje různá API, knihovny a podporu pro návrhové vzory, které programátorovi usnadňují práci a pomáhají s návrhem nového softwaru.
Dříve tady byl AngularJS, ze kterého Angular, jaký známe dnes, vychází. První verze Angular 2.0 vyšla 14. září roku 2016. K 1. dubnu 2019 byla nejnovější verze 7.2.11. Angular je dnes běžně označovaný jako Angular 2+ nebo taky Angular 2 a výše. Je tomu tak, jelikož přepsáním AngularJS došlo k vytvoření nové verze Angular 2, kde se již zkratka „JS“ nenachází a tak docházelo ke zmatkům. Tým programátorů později uvedl, že AngularJS se odkazoval na všechny verze do verze 2 a všechny verze výše než verze 2 se již nazývají Angular 2 a výše, aby nedocházelo ke zmatkům. [8]
Cílem Angularu je přinést nové nástroje a možnosti, které mají napomáhat ke tvorbě, testování a údržbě bohatých a složitých webových aplikací. Angular umožňuje rozšířit HTML kód pomocí vlastních tagů a elementů, pomocí kterých lze využít možnosti aktualizování dat přímo v DOMu. K používání Angularu se váže podpora užívání vzoru Model-View-Controller (dále jen MVC), kde se na straně front-endu nachází Angular a ze serveru, který se nachází v back-endu, přichází data, které Angular poté zobrazuje uživateli.

4.1 Druhy webových aplikaci

Angular při navštívení stránky načítá všechny data najednou – to znamená, že musí zkompilovat všechny HTML elementy, data, komponenty a načíst všechny jiné stavební bloky Angularu. Cílem je provádět toto inicializační nastavení všech komponentů a dalších bloků co nejméněkrát a zajistit co nejvyšší dosažitelný výkon pro uživatele při práci v dané aplikaci. V širším slova smyslu jsou zde dva druhy webových aplikací[8, 6]

1. **Round-trip**
   Takový druh webové aplikace funguje na způsobu vyžádání si počátečního HTML dokumentu ze serveru a při každé interakci uživatele s webovou stránkou, která způsobila změnu URL na novou adresu, si aplikace vyžádá nový HTML dokument. Nevýhodou tohoto druhu webové aplikace je, že uživatel při každé změně adresy URL a načtení nového dokumentu HTML musí čekat, než se stránka aktualizuje.

2. **Single page application**
   V překladu jednostránková webová aplikace na rozdíl od výše zmíněného druhu webové aplikace má k vykreslování obsahu jiný přístup. Počáteční HTML dokument je zaslán

do prohlížeče uživatele, ale interakce uživatele již nevede k vyžádání nového HTML dokumentu ze serveru, nýbrž vyžádání Ajax požadavků pro menší části HTML dokumentu nebo dat, které jsou vloženy do již existujících a uživateli zobrazených elementů. Počáteční HTML dokument tedy není nikdy nahrazen jiným, ale pouze jeho části jsou postupně nahrazovány podle toho, co uživatel na stránce dělá a potřebuje zobrazit.

4.2 Z čeho se Angular skládá?

Angular se po přepsání na verzi 2 dočkal mnoho vylepšení oproti svému předchůdci AngularJS. Základními stavebními bloky Angular webové aplikace jsou komponenty a services (služby). Komponenty i služby jsou jen TypeScriptové třídy, kterým přiřazujeme dekorátor. Podle tohoto dekorátoru hlavní kořenový modul pozná, zda-li se jedná o službu nebo komponent. Příklad těchto dekorátorů lze vidět na výpisu číslo 5.

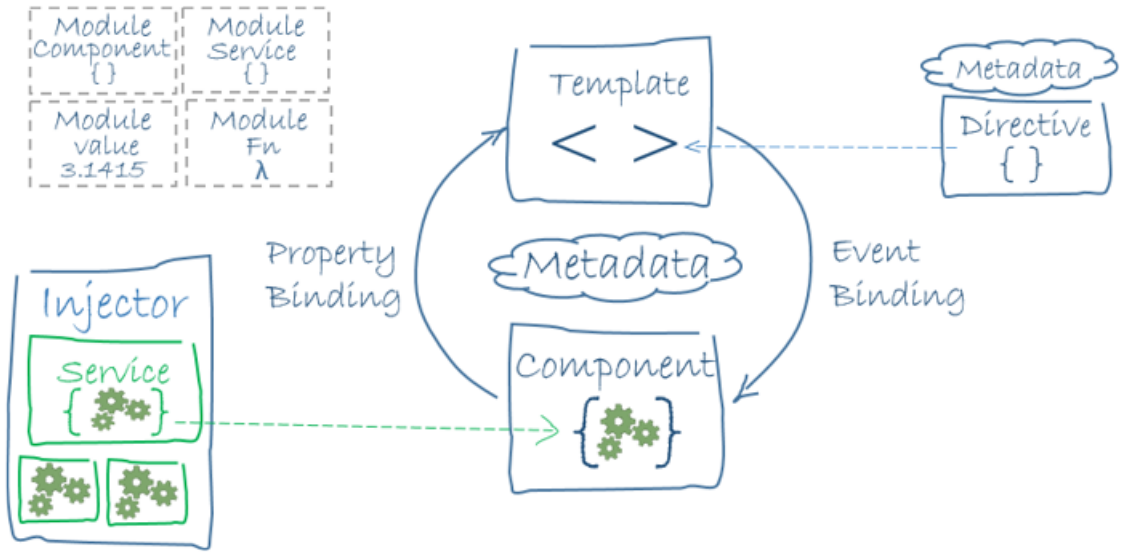
1. Komponenty obsahují logiku a data pro daný komponent. Také obsahují HTML a CSS dokumenty pro popis, jak bude tento komponent graficky podán uživateli se správnými daty, který s ním bude dále pracovat ve webové aplikaci.

2. Komponenty jako takové využívají služby, které poskytují rozšiřující funkční logiku komponentů, nicméně s pohledy komponentu přímo nesouvisejí. Služby mohou být vloženy do komponent, ve kterých jsou potom využívány jejich funkce.

```
...
@Component({
    selector: "my-app",
    styleUrls: "app/app.component.css"
    templateUrl: "app/app.component.html"
})
...
```

Výpis 5: Dekorátor pro komponent

Každá Angular aplikace má dále alespoň kořenový modul, který je nazýván AppModule, který poskytuje možnost zahájit aplikaci. Obecně se v takové webové aplikací využívá těchto modulů mnoho. Například aby webová aplikace mohla využívat služeb routeru (popsán níže), musí programátor takový modul importovat do hlavního modulu. Stejně jako má každá Angular aplikace kořenový modul, tak obsahuje nejméně jeden komponent, který je kořenový. Každý další komponent je spojen s HTML šablonou která definuje pohled, který bude zobrazen uživateli. Jeden z modulů nazvaný Router (nebo taky Směrovač v překladu) poskytuje službu pro navigaci celou webovou aplikací pomocí URL adresy. Zobrazuje příslušné komponenty a pohledy související s danou URL adresou. Také pomocí tlačítek či odkazů na stránce odkazuje na jiné stránky.
Na obrázku níže lze vidět komunikace mezi službami, komponenty a jejich šablonami.



Obrázek 1: Architektura angularu[14]

4.3 Datové vazby v komponentech

Datové vazby v komponentech lze chápat jako propojení mezi elementy HTML šablon komponentů a jejich korespondujícími daty a kódem obsažené v TypeScriptovém souboru daného komponentu. Tímto způsobem lze dynamicky měnit jak data v šablonách HTMl, tak ve zdrojovém kódu, kde se dále s těmito daty pracuje. Datové vazby jako takové se dělí na dva typy:

1. **Jednosměrné datové vazby**
   Anglicky taky nazvané jako *One-Way data binding* jsou jednosměrné datové vazby. To znamená že data proudí pouze jedním směrem.

2. **Obousměrné datové vazby**
   Anglicky nazvané jako *Two-Way data binding*. Oproti jednosměrným datovým vazbám dokáží ty obousměrné zasílat data v obou směrech, ať už je to mezi šablonou a komponentem nebo naopak. Výhodou této datové vazby je, že lze použít při změně dat v šabloně například při vyvolání události. Tento typ datové vazby se používá nejčastěji a především při vytváření formulářů.

4.4 Angular direktiva

Framework Angular dovoluje v HTML šablonách použít přídavné elementy tohoto frameworku. Jedná se například o direktiva *ngFor, kdy danému direktivu přiřadíme zdroj dat (například pole v tomto případě) a Angular se postará o zobrazení značek u kterých jsme direktivum přidali. Dalším direktivem je například *ngIf. Chová se jako klasická podmínka. Pokud je splněna, tak je daná značka zobrazena uživateli, pokud ne, tak nikoliv.

Základní direktiva, se kterými je možné se při vývoji v Angularu setkat jsou:

Tabulka 3: Základní direktiva frameworku Angular[8]

Název direktiva	Popis
ngClass	Používá se pro přiřazení třídy HTML elementům
ngStyle	Používá se pro nastavení individuálních stylů HTML elementům
ngIf	Používá se jako klasická podmínka pro zobrazení jednotlivých HTMl elementů, přesněji popsáno výše
ngFor	Používá se pro vložení obsahu do HTML pro každý prvek obsažený ve zdroji dat
ngSwitch	Používá se pro rozhodnutí/výběr mezi elementy HTML podle hodnoty předem určeného výrazu
ngTemplateOutlet	Používá se pro opakování bloku obsahu v HTML šabloně

4.5 Angular závorky

V Angularu máme více typů závorek, které se používají každá pro jinou operaci nebo pouze čisté zobrazení dat ze zdrojového kódu. Hranaté závorky (závorky [a]) v Angularu značí, že se jedná o jednosměrnou datovou vazbu. Nicméně pokud jde o obecnou nebo direktivu známou Angularu, lze tyto hranaté závorky vynechat.[8]
Hranaté závorky ovšem nejsou jediným typem závorek, které Angular zná. V tabulce číslo 5 lze tedy najít rychlý popis známých závorek pro Angular, jejich význam a kde je možné se s nimi setkat.

Tabulka 4: Typy závorek Angular[8]

Výraz	Popis
[cíl] = "výraz"	V tomto případě jde o jednosměrnou datovou vazbu, kde data proudí z výrazu do cíle
{{ výraz }}	Vazba HTML šablony a zdrojovým kódem, kde do HTML šablony jsou přiřazena daná data dle výrazu
(cíl) = "výraz"	Jedná se o jednosměrnou datovou vazbu, která se používá pro zpracování události. Příkladem může být událost kliku na tlačítko. Například *(click)="myFunction()"*
[(cíl)] = "výraz"	Tato kombinace značí oboustranou datovou vazbu

4.6 Šablony a styly komponentů

Ve výpise číslo 5:*Dekorátor pro komponent* si lze všimnout parametrů *styleUrls* a *templateUrl*. Pro šablony v HTML formátu používáme templateUrl.

HTML je značkovací jazyk používaný při tvorbě webových stránek. Byl založen roku 1990 a dnes je k dispozici verze HTML5. Soubory psané v jazyce HTML se vyznačují množinou značek (nebo taky tagů) a vlastností. Názvy značek a jejich vlastností se píší mezi úhlové značky < a >. Pokud je značka párová, tak je potřeba aby byla po otevření i uzavřena. Taková to značka lze uzavřít použitím lomítka ihned za úhlovou značkou <. Jako příklad poslouží například značka div v dalším výpise.[9]

```
<div class="myDiv"> //Otevření párové značky
   Text
</div> //Uzavření párové značky
```

Výpis 6: Příklad párové značky v jazyce HTML

Jednotlivým značkám lze přiřazovat unikátní ID nebo třídy, přes které lze tyto značky stylizovat odkazováním v CSS souboru. Jednotlivé značky lze jednoduše stylizovat i pomocí vlastnosti *style* a není potřeba jej stylizovat v samostatném CSS souboru.

CSS soubory neboli kaskádové styly určují a popisují zobrazení elementů na webových stránkách vytvořených pomocí jazyka HTML. V dnešní podobě známe již verzi CSS3, která je plně kompatibilní s HTML5. Hlavním úkolem kaskádových stylů je oddělit strukturu a obsah webu od jeho vzhledu. Obsah webové stránky je tedy psán v HTML souboru a rozmístění po stránce, přidání barev a obecně změně stylu dochází pomocí kaskádových stylů. Takovýchto kaskádových stylů lze použít v šabloně vícero. Angular dokonce umožňuje použití kaskádových stylů z jiného komponentu. Výhodou tohoto použití je pokud mají oba komponenty a jejich šablony téměř totožný obsah - není zde potřeba psát kaskádové styly ve stejné podobě jako u předchozí komponenty.[9]

4.7 Získávání dat ze serveru pro klienta

Logika, kterou služby a komponenty dále využívají je často rozdělena mezi klient a server. Server jako takový obsahuje veškerou logiku pro správu dat, jako další obsahuje trvalé úložiště těchto dat, kde se typicky jedná o databázi. V případě použití SQL databáze server obstarává veškerou logiku jak se k těmto datům přistupuje, jak je lze získat a stará se o jejich správů. Mezi tuto správu například patří provádění SQL dotazů nad danou databází a zpracování výsledků do podoby, aby mohly být posléze odeslány klientovi k zobrazení pro uživatele. Je to rozděleno z jednoho prostého důvodu. Klient by neměl přistupovat přímo k úložišti dat, ale využívat na zobrazení a odesílání dat server, jelikož nedojde k těsnému spojení klienta a dat. V praxi by to ztěžovalo měnit data v úložišti dat a přitom provádět změny v klientovi. Klient si pouze vyžádá

data, ale o tom, jak jsou získávány nebo zpracovány není informován. Jeden z běžných způsobů získávání dat ze strany serveru je například pomocí HTTP požadavků.

Tabulka 5: Tabulka HTTP operací obecně používaných pro získání dat ze serveru[4, 5]

Operátor	Význam a popis
GET	Získává data podle specifikace v URL adrese
PUT	Aktualizuje data podle specifikace v URL adrese
POST	Přidává data, typicky z formulářů
DELETE	Maže data

1. Metodou GET bychom měli data pouze získávat a nikoliv je jakkoliv měnit. Prohlížeč očekává, že požadavky GET bude moci opakovaně používat bez toho, aby měnil data na serveru.

2. Metoda PUT se používá pro aktualizaci dat v databázi podle specifikovaného kritéria v URL. Například použitím PUT metody, kde pošleme v požadavku objekt uživatele, tak dojde k jeho aktualizaci v databázi.

3. Metoda DELETE je podle specifikovaného kritéria provedena pouze jednou a v dalších voláních již v databázi beze změny. Například pokud smaže objekt uživatele, který byl specifikován, tak při druhém volání již nemá co smazat. Toto samozřejmě neplatí, pokud jiný klient daný objekt opět nevytvoří.

4.8 Shrnutí kapitoly

V této podkapitole se nachází shrnutí pomocí málo pár odrážek vše, co je popsáno v kapitole o frameworku Angular.
- Angular je open-source framework založen na programovacím jazyku TypeScript
- Framework jako takový obsahuje již předem naprogramované knihovny a API, které lze využívat při vývoji.
- Značení Angular 2 se vztahuje na všechny verze Angularu 2 a výše.
- Cílem Angularu je přinést nové nástroje a možnosti při tvorbě bohatých a složitých webových aplikací.
- Webové aplikace dělíme na dva druhy webových aplikací. Mezi tyto dva druhy patří Round-trip a Single page application.
- Každá Angular webová aplikace se skládá z root modulu, taktéž nazvaném App Module, komponentů, služeb a dalších programátorem definovaných modulů.
- Angular umožňuje použití přídavných direktiv tohoto frameworku u jednotlivých tagů v HTML šablonách.

- Angular jako takový podporuje více typů závorek, kdy každá kombinace znamená pro Angular vykonání jisté operace.
- Pokud je webová aplikace rozdělena na frontend a backend, tak jsou data získávány především ze serveru pomocí HTTP požadavků.

5 Návrh systému

Systém je navržen tak, aby byl intuitivní a pro uživatele co nejjednodušší na pochopení. Popis použitých technik a nástrojů je popsán v dalších podkapitolách, stejně jako rozdělení uživatelů v rámci webové aplikace a aplikace samotné a jejího navržení jak po funkční tak vzhledové stránce.

5.1 Použité nástroje

Na straně backendu (serveru) je použit ASP.NET, jež je součástí .NET frameworku vyvíjen firmou Microsoft, který obsahuje nejnovější techniky a myšlenky pro vývoj webových aplikací a služeb. Výhodou ASP.NET je, že programátoři mohou vyvíjet systémy v jakémkoli jazyce, který podporuje CLR(Common Language Runtime), jako jsou například Visual Basic .NET a C#. Pro implementaci tohoto projektu byl použit programovací jazyk C#. Jelikož C# vyvíjí taktéž Microsoft, tak podpora vývojových prostředí jako je Visual Studio a Visual Studio Code je velmi intuitivní a na vysoké úrovni.
Pro ukládání dat je na straně serveru použitý Microsoft SQL Server.
Na straně frontendu (klienta) je použitý framework Angular, který se bude starat o výslednou grafickou podobu a podávání správných informací ze serveru uživateli. Co je to Angular, z čeho se skládá nebo například jak komunikuje se serverem jsem popsal v kapitole číslo 4. *Představení frameworku Angular*.
Pro grafickou podobu celého systému byla použita knihovnu Angular Material. Tuto knihovnu vyvíjí a spravuje opět Google a jelikož Angular je také víceméně jeho projekt, tak integrace material designu do Angularu je již podporována. Knihovna Angular Material pomáhá celé webové aplikaci držet konzistentní design v tom smyslu dodržet ve všech komponentech stejné odstíny předem stanovených barev, které se používají například na tlačítka, navigační menu nebo třeba vstupní pole. Další výhodou použití této knihovny je podpora animací různých prvků. Ať už jde o vizuální zpětnou vazbu pro uživatele při provedení akce nebo třeba přechody mezi jednotlivými prvky. Další výhodou je možnost přepínat mezi denním a nočním režimem, kdy dojde k výměně barev pozadí. Tato možnost zahrnuje taktéž změnu primárních a sekundárních barev, které jsou předem nadefinované. Mezi výhody této knihovny jistě patří i možnost integrace Material Icons. Jedná se o ikony, které vyvíjí opět Google a obsahuje je například operační systém Android. Použití těchto ikon ve webové aplikaci přidává grafickou podporu a zvyšuje tak pochopení například tlačítka či odkazu.

5.2 Ověření

Ověření uživatelů při přihlašování do systému probíhá pomocí JSON Web Tokenů (zkratkovitě je to JWT). Jedná se tedy o ověření pomocí tokenů, které jsou při validním přihlášení uživatelů generovány a ukládány do lokálního úložiště daného uživatele. Pomocí JSON Web tokenů lze bezpečně přenášet data mezi serverem a klientem v podobě JSONu. JSON je popis JavaScripto-

vého objektu, jež je formát pro ukládání a přepravu dat na internetu. Výhodou takového JSON Web Tokenu je to, že může být podepsaný a tudíž se dá ověřit jeho pravost jako například pokud byl vydán tím samým serverem.[3]
Zmíněný token potom uživatel posílá v hlavičce při každém HTTP požadavku, kde dochází k jeho ověření, zda-li může daný uživatel požadavek po serveru chtít. Pokud token není správný nebo nelze ověřit, systém odpoví že daný uživatel není oprávněný ve vykonání požadované akce. V implementaci systému jsou tokeny ověřovány pouze z lokálního úložiště uživatele, který obsahuje podpis a bezpečnostní kód pro přístup k přenášeným datům. Token má také nastavenou dobu, za kterou platnost daného tokenu vyprší a po této době se stává nepoužitelný z hlediska ověřování. Daný uživatel se po této lhůtě bude muset znovu přihlásit a ověřit.
Uživatelské účty se dělí na 3 skupiny:

1. **Administrátor**
   Jsou to uživatelské účty, které mohou v aplikaci provádět téměř cokoliv. Jsou to účty stvořené pro správu daného systému.

2. **Uživatel**
   Účet, který by měli mít normální uživatelé, kteří mohou přidávat projekty, úkoly a bugy, ovšem nemají právo sahat na seznam uživatelů či nové uživatele přidávat.

3. **Uživatel jen pro čtení**
   Je to účet, ve kterém jeho uživatelé mohou systém jen procházet a nemají žádná práva jakkoliv editovat, co je v něm uloženo. Takový uživatel nemůže založit projekt a k němu přiřazovat úkoly nebo bugy. Také nemůže měnit jakkoliv seznam uživatelů.

Hesla uživatelských účtů nejsou v databázi uloženy jen jako čistý text. Všechny hesla procházejí hashovacím algoritmem SHA256 a až poté jsou uloženy do databáze. Při přihlášení se tedy porovnává hash uložený v databází s hashem vygenerovaným při zaslání přihlašovacího formuláře.

5.3 Architektura MVC

Tento systém je postaven na architektuře MVC (Model-View-Controller). Tento vzor vznikl koncem 70. let. Vzor popisuje jak organizovat částí grafického rozhraní, logiku získávání dat a interakci uživatele s webovou aplikací. Respektive vzor MVC popisuje jak oddělit model domény a logiku řadiče od uživatelského rozhraní – to HTTP požadavkům vyhovuje, které jsou vlastně základem pro webové aplikace. Oddělením HTML kódu od zbytku aplikace podle tohoto vzoru má za důsledek, že testování a údržba lze provádět jednodušeji, než kdyby byl HTML kód obsažen dohromady se zbytkem aplikace. [3, 4]
Ve zkratce to znamená, že systém bude rozdělen na tři části:

1. Model, který obsahuje nebo reprezentuje podobu dat, se kterými budou uživatelé pracovat a které jim budou zobrazovány. Modely dále rozdělujeme na dva typy. Jako první to jsou

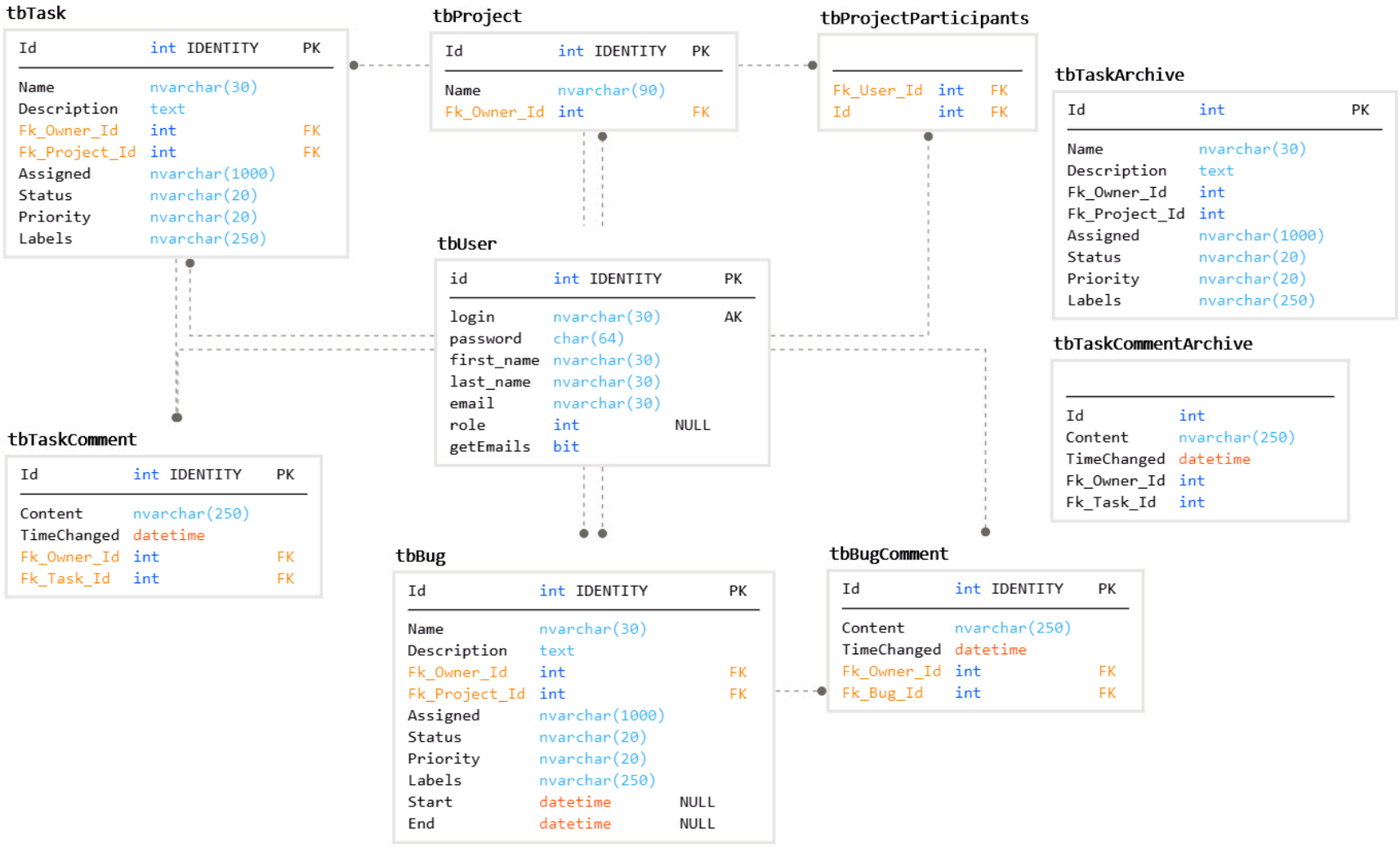
pohledové modely, které jen jednoduše reprezentují podobu dat přenášených mezi kontrolery a pohledy. Jako druhé to jsou doménové modely, které nad rámec reprezentace dat jako v minulém případě, také obsahují operace, pravidla a různé transformace prováděné nad těmito daty. Nicméně nezatěžují se získáváním těchto dat a zobrazováním uživatelského rozhraní.

2. Jako další částí jsou tady pohledy, které se využívají pro vykreslení a prezentaci dat různých modelů tak, aby byly uživateli podávány relevantní informace. Jsou součástí uživatelského rozhraní. Obsahují tedy logiku potřebnou pro zobrazení požadovaných elementů uživateli. Obdobně jako modely se žádnou jinou logikou nezatěžují.

3. Třetí a poslední částí vzoru jsou kontrolery. Ty mají za úkol zpracovávat požadavky, provádět operace nad modely a vybírat pohledy, které budou zobrazeny uživateli.

5.4 Databázový kontext

V této podkapitole budou popsány třídy, ve kterých se pracuje s databází pro ukládání dat z celé webové aplikace. Je zde obsažená veškerá logika pro komunikaci s databází jako jsou CRUD operace. Jako ORM (objektově relační mapování) je používán framework Dapper, který pomáhá mapovat výstup dotazu na C# třídy. Jde o vysoce výkoný framework vytvořený týmem StackOverflow a vydán jako open-source. Pracuje tedy s modely a jednoduchými příkazy pracuje s databází. Oproti psaní vlastního ORM je Dapper na vysoké úrovni z pohledu bezpečnosti a výkonu. [13] Každá třída obsahuje metody, ve kterých se nachází ať už kód pro CRUD operace nebo například řetězce pro editaci jen vybraných vlastností dané tabulky. Připojovací řetězec je získáván z konfiguračního souboru app.json, ve kterém je uložen na bezpečném místě.
Jako databázový server byl použitý Microsoft SQL Server (MSSQL). Hlavní tabulkou je v celé databázi tbUser (uživatel), na kterou se poté vážou téměř veškeré ostatní tabulky. Jediné tabulky, které nemají cizí klíč právě Id uživatele jsou tabulky, které se používají pro archív úkolů a jejich komentářů.
Pro mazání projektů, úkolů, bugů a uživatelů jsou používány uložené procedury na SQL serveru, jelikož dochází k mazání záznamů z více tabulek než jedné.



Obrázek 2: Model databáze

5.5 Modely

V podkapitole číslo 3. Databázový kontext byl popsán framework Dapper. Tento framework dále využívá modely, které stačí přiřadit k danému dotazu, který Dapper provádí. Jelikož každý model má v sobě atribut [Table(název tabulky)], tak lze daný model jen přiřadit a Dapper ví, do které tabulky má data vkládat/editovat/mazat. Dapper kromě atributu Table nabízí další atributy jako například [Key], který lze vložit na řádek nad proměnnou, díky kterému Dapper ví, že se jedná o hlavní klíč tabulky. Dalším argumentem je argument [Column(název sloupce)].[13] Tímto způsobem může mít model jinak pojmenované proměnné, než jsou v databázi. Každý model taktéž může mít proměnné, které v databází vůbec nejsou a Dapper je může ignorovat. Pro takový případ Dapper nabízí atribut [Editable(false)]. Příklad je uveden níže:

```
[Table("tbProject")]
public class ProjectModel
{
   [Key]
   public int Id { get; set; }
   public string Name { get; set; }
   [Column("Fk_Owner_Id")]
   public int Owner_Id { get; set; }
   public string Assigned { get; set; }

   //Vlastnosti, které nejsou v databází a jsou ignorovány
   [Editable(false)]
   public string ParticipientsString { get; set; }
}
```

Výpis 7: Ukázka atributů frameworku Dapper

5.6 Kontrolery

Kontrolery se zabývají a řeší příchozí požadavky ze strany uživatele z frontend aplikace. Každou veřejnou metodu v kontrolerech lze vyvolat z webu pomocí předem stanovené URL. Ke každé této metodě lze přidat různé atributy. Hlavním atributem je zajisté jakou metodu HTTP protokolu daná metoda použije (HTTP metody jsou přesněji popsány v kapitole Angular – *2.3 Získávání dat ze serveru*). Jako další hlavní atribut lze určitě považovat atribut [Route(url)], jelikož daný argument definuje URL adresu, přes kterou uživatel metodu vyvolá. Dalším atributem je atribut [Authorize], který přijímá jako parametr řetězec, například v tomto systému, předem stanovených rolí uživatelských účtů. To znamená, že pokud u metody bude atribut [Authorize(admin)], tak danou metodu bude moci vyvolat pouze uživatelský účet, který danou rolí

disponuje.

```
[HttpGet] //Metoda HTTP protokolu
[Authorize(Roles="admin, editableUser, readOnlyUser")]
[Route("api/Users/Fetch")]
public IEnumerable<User> Fetch()
{
   return userObject.GetUsers();
}
```

Výpis 8: Ukázka metody z kontroléru

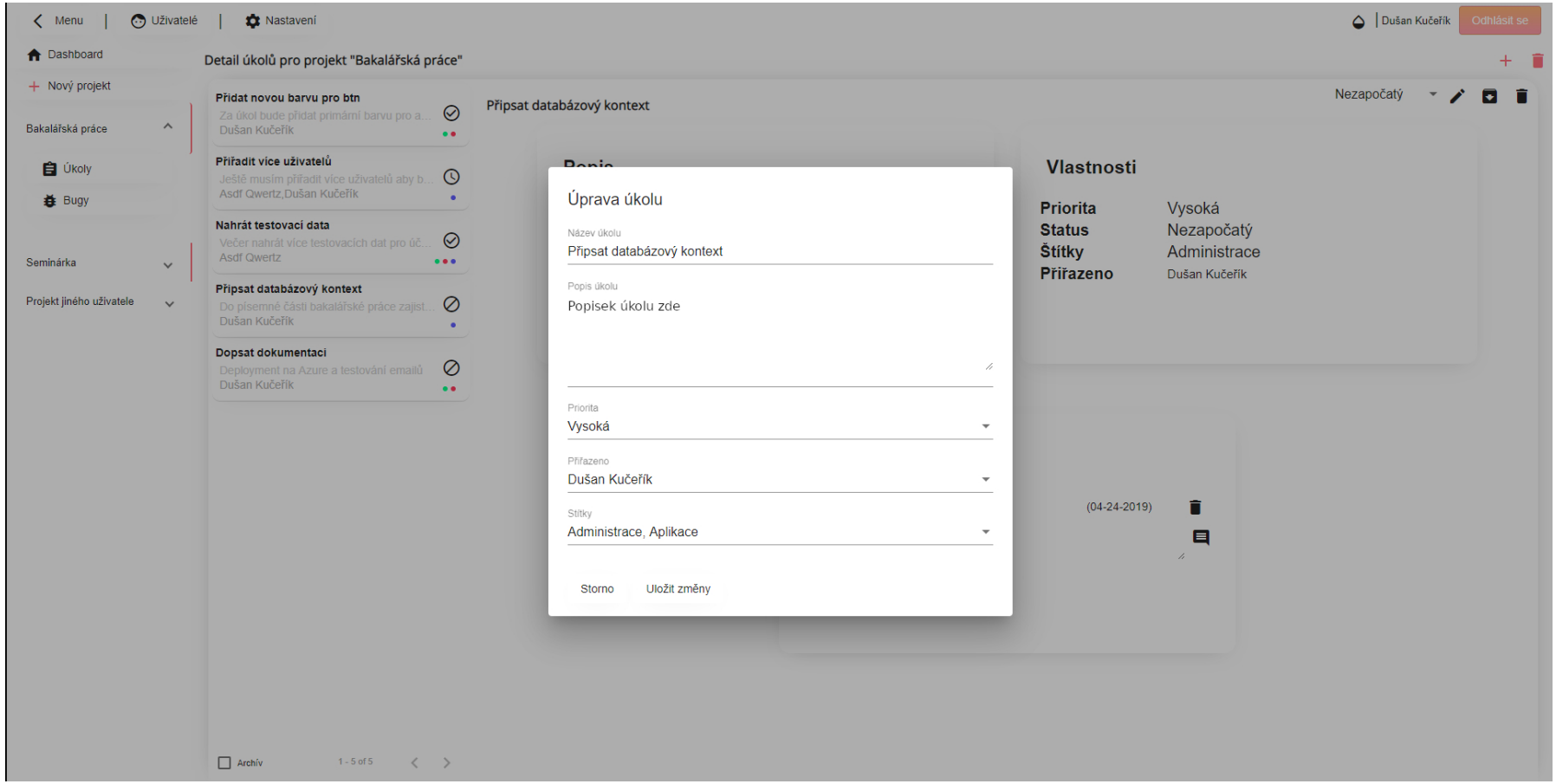
5.7 Emailový klient

Systém dokáže upozornit uživatele emailem, kdykoli je jeho uživatelský účet přiřazen k projektu. K tomuto účelu byla použita knihovna multiplatformního poštovního klienta MailKit postavena na C# knihovně MimeKit.[12] Použití této knihovny je velmi intuitivní a jednoduché. Pro odeslání emailu stačí vytvořit instanci MimeMessage, které poté definujeme různé vlastnosti. Mezi tyto vlastnosti spadá zajisté emailová adresa odesílatele, emailová adresa příjemce, předmět a textový obsah emailové zprávy.
Nicméně pro úspěšné odeslání emailu musí být poskytnut ověřený SMTP server, ze kterého bude odcházející pošta odesílána.

5.8 Angular aplikace

Skrze celou webovou aplikaci zůstává na horní straně navigační lišta, kde lze jednoduše přistupovat na domovskou stránku, do nastavení účtu, odhlášení přihlášeného uživatele nebo třeba přepínač mezi tmavým a světlým motivem. Webová aplikace disponuje dvěmi barevnými motivy, kde každý obsahuje 3 barvy. Obsaženy jsou barvy primární pro navigační menu, sekundární pro doplňky jako jsou například tlačítka, kalendář, podtrhávání a podobné a jako poslední barva upozornění. Odstín barvy upozornění zůstává u obou stejný. Barvy jsou vybrány podle Google Material palety tak, aby šly správně a jednoduše vidět na tmavém či světlém pozadí. Jako výchozí motiv aplikace je nastaven světlý motiv, který lze jednoduše přepínačem přepnout. Na levé straně aplikace se nachází boční menu, kde se zobrazují jednotlivé projekty přihlášeného uživatele a také odkaz pro přidání dalšího projektu. Celé boční menu lze schovat pomocí tlačítka a opět zobrazit. Při schování bočního menu dochází k roztažení zbývajícího obsahu na celou obrazovku.
Celá webová aplikace má jednotný design s animacemi díky frameworku Angular Material. Skrze celou aplikaci provází uživatele prvky stínů pod kartami a dalšími elementy. Na hlavní domovské obrazovce se nacházejí karty s rychlým přístupem k seznamu uživatelů, vytvoření projektu

a další.
Pro kontrolu zda-li je uživatel ověřen a stále má platný JWT token zde slouží třída AuthenticationGuard. Jde o třídu, jež implementuje rozhraní CanActivate pomocí kterého se daná metoda provede při každé změně URL. Zde probíhá kontrola a pokud jí uživatel nesplňuje, tak je odkázán na přihlašovací obrazovku, kde se musí znovu přihlásit a ověřit jako platný uživatel.
Pro komunikaci se serverem zde slouží služby. Každá metoda pouze volá danou API z kontroleru ze serveru, ke které při každém volání také posílá JWT token pro kontrolu, zda-li je uživatel oprávněn danou metodu použít.
Úkolům a bugům v projektech lze přiřadit štítky, které lze poté vidět v bočním menu pro rychlý přehled. Taktéž lze přiřadit status ve kterém stádiu se úkol či bug zrovna nachází. Každý status má v bočním menu přiřazenou svou ikonu, sloužící stejně jako u štítků k rychlému přehledu. Daným úkolům a bugům je možné přiřadit uživatele, kteří na nich mají pracovat.



Obrázek 3: Design webové aplikace

5.9 Shrnutí návrhu systému

- Systém je navržen pro co nejjednodušší použití a rychlé pochopení ze strany uživatele.
- Na straně backendu je použit ASP.NET, jež je součástí .NET frameworku.
- Na straně frontendu je použitý framework Angular, který má za úkol získávat data ze strany serveru a následně je graficky prezentovat uživateli v podobě webové aplikace.
- Pro grafickou podobu je použitá knihovna Angular Material, jež následuje pokyny pro design od Google Material Design.
- Pro snadnější pochopení uživatelského prostředí je ve webové aplikaci integrováný systém Material Icons.
- Pro ověřování uživatelů v celém systému slouží časově omezené JWT tokeny uložených v lokálním úložišti uživatele.
- Komunikaci a veškeré získávání, přidávání a aktualizace dat probíhá přes ORM Dapper, jež usnadňuje práci a dělá komunikaci s SQL bezpečnější.
- Emailovou komunikaci zajišťuje použití knihovny MailKit postavené na C# knihovně MimeKit.

6 Závěr

V písemné části práce bylo popsáno 5 vybraných systémů, mezi které byly vybrány Wrike.com, monday.com, ActiveCollab, Zoho Projects a Asana. Všechny tyto systémy byly aktivně a pečlivě odzkoušeny. Každý systém byl důkladně popsán, ať už se jedná o stránku grafické podoby nebo například nabídce vymožeností, které ostatní systémy nenabízejí.
V další části práce byl představen jazyk TypeScript. V tomto představení programovacího jazyka byly popsány základní části, nikoliv specifické detaily pro pokročilé programátory. Mezi tyto základní části byly zařazeny například proměnné, podporované datové typy, třídy a jejich konstruktory a modifikátory přístupu, generika a operátory jazyka TypeScript.
Další části byl popsán a představen framework Angular založený na programovacím jazyku TypeScript. Tato kapitola obdobně jako kapitola předešlá obsahuje základní rysy a znaky frameworku Angular pro tvorbu webových aplikací. Ať už se jedná o části, ze kterých je Angular složen, použité direktiva, typy závorek nebo třeba šablony a jejich styly při použití v komponentech. Jelikož se jedná o webovou aplikaci, která má uložené data na server, byla zde popsána i možnost získávání dat ze serveru za pomocí HTTP požadavků.
V poslední kapitole pojednávající o systému byl popsán návrh celého systému a použitých technologií a nástrojů při jeho implementaci. Na straně serveru byla použita technologie ASP.NET pro správu zdrojového kódu a pro ukládání dat byla použita databáze Microsoft SQL Server. Na straně uživatelského rozhraní byl použit framework Angular, jež se stará o podání dat v podobě srozumitelné pro uživatele v kombinaci s Angular Material pro jednotný design skrze celou webovou aplikaci. Dále v byly v této kapitole popsány nástroje pro odesílání emailů, ověření uživatelů při přihlášení a Angular aplikace z pohledu rozmístění elementů v uživatelském rozhraní a její logiky.

Literatura

[1] MAHARRY, Daniel. TypeScript revealed. California: Apress, [2013]. Expert's voice in .NET. ISBN 978-1-4302-5725-7.

[2] FENTON, Steve. Pro TypeScript: application-scale JavaScript development. New York, N.Y.: Apress, [2014]. ISBN 978-1-4302-6791-1.

[3] FREEMAN, Adam. Pro ASP.NET MVC 5. Fifth edition. New York, New York: Apress, [2013]. Expert's voice in ASP.NET. ISBN 978-1-4302-6529-0.

[4] UGURLU, Tugberk, Alexander ZEITLER a Ali KHEYROLLAHI. Pro ASP.NET web API: HTTP web services in ASP.NET. Berkeley, CA: Apress, c2013. Expert's voice in .NET. ISBN 978-1-4302-4725-8.

[5] FREEMAN, Adam. Pro AngularJS. New York, NY: Apress, [2014]. ISBN 978-1-4302-6448-4.

[6] FRISBIE, Matt. Angular 2 Cookbook. Birmingham: Packt Publishing, 2017. ISBN 978-1-78588-192-3.

[7] WILLIAMSON, Ken. Learning AngularJS: a guide to AngularJS development. Sebastopol, CA: O'Reilly Media, 2015. ISBN 978-1-491-91675-9.

[8] FREEMAN, Adam. Pro Angular. 2nd. United Kingdom: Apress, 2017. ISBN 978-1-4842-2306-2.

[9] DUCKETT, Jon. HTML and CSS: design and build websites. Indianapolis, Indiana: John Wiley and Sons, 2014. ISBN 978-1-118-00818-8.

[10] Capterra System Comparison. Capterra [online]. Dostupné z: https://www.capterra.com/project-management-software/compare/114491-147657-76113-169455/ActiveCollab-vs-monday-com-vs-Wrike-vs-Zoho-Projects

[11] Capterra System Comparison. Capterra [online]. Dostupné z: https://www.capterra.com/project-management-software/compare/184581-147657-76113/Asana-vs-monday-com-vs-Wrike

[12] MimeKit MailKit. Mimekit [online]. Dostupné z: http://www.mimekit.net/docs/html/Introduction.htm

[13] Dapper. Dapper Documentation [online]. Dostupné z: https://dapper-tutorial.net/dapper

[14] AngularOverview. In: Angular Docs [online]. [cit. 2019-04-24]. Dostupné z: https://angular.io/generated/images/guide/architecture/overview2.png